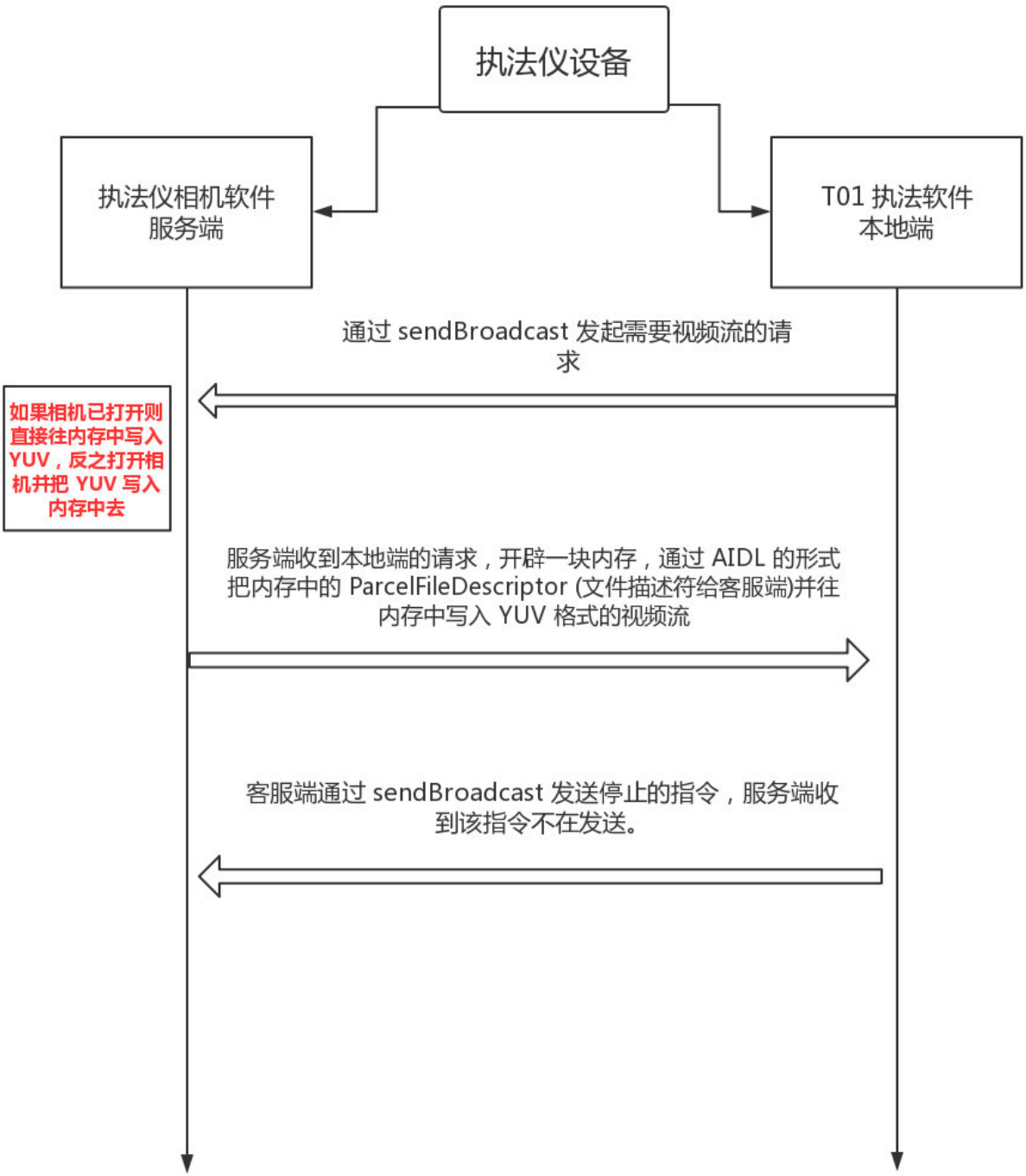
执法仪相机视频流传输协议说明

简介说明

由于项目需求，需要在执法仪本地录像的时候，执法软件能正常的使用设备本身的 Camera 资源。由于 Android 系统自身不允许多个软件同时使用 Camera 资源，故开发一套**内存共享**子码流传输协议，当执法软件需要视频流的时候，向执法仪设备请求往 MemoryFile 中写入 YUV 格式的视频流，执法软件每隔一段时间循环的去指定的内存中 read YUV 视频流。

完整代码入口

流程图



效果图

- 本地相机未打开

  效果图未显示

- 本地相机打开

  效果图未显示请点击这里

简要流程及示例代码说明

1. 执法仪服务端收到客户端的开启视频流写入内存的指令

```
    //处理客服端发送过来的需要子码流的数据
    private void onHandleAction(Context context, Intent intent) {
        switch (intent.getAction()) {
            /**
             * 需要子码流
             */
            case Constants.ACTION_CAMERE_CORE_SHOW:
                //如果正在发送视频流，就不需要执行后面代码了
                if (!MemoryFileServiceManager.getInsta(context).isSendVideoFrame())
                    MemoryFileServiceManager.getInsta(context).setSendVideoFrame(true,
intent);
                break;
        }
    }
```

2. 服务端获取开启视频的条件

```
    private void sendVideoFrame(Intent intent) {
        if (intent != null && intent.getExtras() != null) {
            Bundle extras = intent.getExtras();
            //获取需要预览的宽
            Constants.PREVIEWHEIGHT = extras.getInt(Constants.Config.PREVIEW_WIDTH,
1280);
            //获取需要预览的高
            Constants.PREVIEWHEIGHT = extras.getInt(Constants.Config.PREVIEW_HEIGHT,
720);
            //需要绑定对方服务的进程
            Constants.BIND_OTHER_SERVICE_PCK =
extras.getString(Constants.Config.BIND_OTHER_SERVICE_PCK, "");
            //需要绑定对方服务的全路径
            Constants.BIND_OTHER_SERVICE_CLASS =
extras.getString(Constants.Config.BIND_OTHER_SERVICE_CLASS, "");
            //需要开启 Camera ID 的前置还是后置 0: 后置 1: 前置
            Constants.CAMERA_ID = extras.getInt(Constants.Config.CAMERA_ID, 0);
        }
    }
```

3. 服务器是否开启相机，如果已经开启则不需要开启

```
        //是否摄像头
        if (mCamera == null)
            openCamera();
```

4. 服务端初始化一块内存，用于写入 YUV 视频流。

```
mMemoryFile = initMemoryFile(Constants.MEMORY_FILE_NAME, Constants.MEMORY_SIZE);
```

5. 绑定对方服务，提供文件描述符号

```
    /**
     * 绑定对方服务，提供 文件描述符
     */
    private void bindOtherService() {
        try {
            if (TextUtils.isEmpty(Constants.BIND_OTHER_SERVICE_PCK) ||
TextUtils.isEmpty(Constants.BIND_OTHER_SERVICE_CLASS))
                throw new NullPointerException("PCK or CLSS is null ?");
            Intent intent = new Intent();
            ComponentName cmp = new ComponentName(Constants.BIND_OTHER_SERVICE_PCK,
Constants.BIND_OTHER_SERVICE_CLASS);
            intent.setComponent(cmp);
            context.bindService(intent, mCameraServiceConnection,
Context.BIND_AUTO_CREATE);
        } catch (Exception e) {
            Log.e(TAG, e.getMessage());
        }
    }
```

6. 绑定对方服务成功，交于文件描述符 ParcelFileDescriptor

```
            mCameraService = ICameraCoreService.Stub.asInterface(binder);
            if (mMemoryFile != null) {
                try {
                    //反射拿到文件描述符号
                    mParcelFileDescriptor =
MemoryFileHelper.getParcelFileDescriptor(mMemoryFile);
                    if (mParcelFileDescriptor != null) {
                        mCameraService.addExportMemoryFile(mParcelFileDescriptor,
Constants.PREVIEWWIDTH, Constants.PREVIEWHEIGHT, Constants.MEMORY_SIZE);

                    }
```

7. 发送数据，当标志位为 byte[0] == 0 代表服务端可将 YUV 写入内存， == 1 ，代表客服端可以读取可用的 YUV 数据。

```
    /**
     * 读标志位 写入视频流
     *
     * @param memoryFile
```

```
     */
    public void writeBytes(MemoryFile memoryFile) {
        try {
            if (mYUVQueue.size() > 0) {
                BufferBean mBufferBean = new BufferBean(Constants.BUFFER_SIZE);
                //读取标志符号
                memoryFile.readBytes(mBufferBean.isCanRead, 0, 0, 1);
                //当第一位为 0 的时候，代表客服端已经读取了，可以正常将视频流写入内存中
                if (mBufferBean.isCanRead[0] == 0) {
                    //拿到视频流
                    byte[] video = mYUVQueue.poll();
                    if (video != null)
                        //将视频流写入内存中
                        memoryFile.writeBytes(video, 0, 0, video.length);
                    //标志位复位，等待客服端读取视频流
                    mBufferBean.isCanRead[0] = 1;
                    memoryFile.writeBytes(mBufferBean.isCanRead, 0, 0, 1);
                } else {
                    Log.d(TAG, "readShareBufferMsg isCanRead:" +
mBufferBean.isCanRead[0] + ";length:"
                            + mBufferBean.mBuffer.length);
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
            sendBroadcast(Constants.ACTION_FEEDBACK, e.getMessage());
        }
    }
```

8. 开启成功或者失败等其他错误消息反馈给客服端

```
//返回给客服端
public void sendBroadcast(String action,String content) {
        Intent intent = new Intent();
        intent.setAction(action);
        ComponentName componentName = new ComponentName("com.t01.sharevideostream",
                "com.t01.sharevideostream.revices.FeedBackReceiver");
        intent.setComponent(componentName);
        Bundle extras = new Bundle();
        extras.putString(Constants.ACTION_FEEDBACK_CONTENT, content);
        intent.putExtras(extras);
        context.sendBroadcast(intent);
    }
```

9. 服务端收到客服端停止写入 YUV 的命令

```
    private void onHandleAction(Context context, Intent intent) {
        switch (intent.getAction()) {
            /**
             * 不需要子码流
             */
            case Constants.ACTION_CAMERE_CORE_HIDE:
        MemoryFileServiceManager.getInsta(context).setSendVideoFrame(false, intent);
                break;
        }
    }
```